MANCHESTER
1824
The University of Manchester

Visual Attention via bio-inspired spiking neural network

Third Year Project

Alexander Angelov

April 2021

BSc(Hons) Computer Science

Supervisor: Prof. Angelo Cangelosi

Acknowledgements

I would like to express my great appreciation to my supervisor, Prof. Angelo Cangelosi for the valuable support throughout such a difficult year under Covid-19. I am very thankful for his helpful suggestions, for the time he spent talking with me and for his ability to motivate me when I was feeling depressed or out of track.

I would also like to thank Alex Rast for pointing me to key resources and for spending time talking to me about the development made in the area of my project and discussing key concepts for neuroscience and spiking neural networks.

Last but not least, I want to thank my family for the support and their encouragements.

Contents

1. Introduction
    1.1. Motivation
    1.2. Objectives
        1.2.1. Learning and exploring
        1.2.2. Development
    1.3. Report Structure
    1.4. Impact of Covid-19
2. Background
    2.1. Visual attention
    2.2. Neuroscience and human visual system
        2.2.1. The neuron, action potential and neurotransmitters
        2.2.2. The retina and the lateral geniculate nucleus
        2.2.3. Visual Cortex and Lateral Intraparietal Cortex
    2.3. Image processing
        2.3.1. The image
        2.3.2. Local normalization
        2.3.3. Kernels
        2.3.4. Gaussian filters
    2.4. Spiking neural networks
        2.4.1. Spike-Timing Dependent Plasticity
        2.4.2. Convolution layer
        2.4.3. Pooling (subsampling) layer
        2.4.4. Winner-take-all technique
3. Methodologies
    3.1. Areal filtering
    3.2. Cognitive architecture of the model
        3.2.1. The process of creating retina like spike trains
        3.2.2. Population
        3.2.3. Cognitive architecture of the spiking neural network
4. Implementation
    4.1. Technology
    4.2. Datasets
    4.3. The implementation of the visual attention model
        4.3.1. Converting images to spike trains
        4.3.2. Tools for creating the spiking neural network
        4.3.3. Model visualization
5. Experiments and results
    5.1. Converting images to spikes experiments
        5.1.1. Input size experiments
            5.1.1.1. Experiment comparing results to initial value
            5.1.1.2. Experiment comparing to the previous result
        5.1.2. Runtime experiments
            5.1.2.1. Experiment comparing results to initial value
            5.1.2.2. Experiment comparing to the previous result
        5.1.3. Experiment comparing image spike trains

5.2. Experiments on the spiking neural network
5.2.1. Replication of the original experiment
5.2.2. Experiments on images with almost no noise
5.2.3. Experiments on real-world images
6. Future development
6.1. Replicating the brain’s visual system in greater detail
6.2. Visual attention in video
6.3. Recognizing the surrounding objects
7. Reflection and conclusion
7.1. Challenges
7.2. Knowledge development
8. Bibliography

List of abbreviations and terms

AV -> Autonomous Vehicles

CNN -> Convolutional Neural Network

GUI -> Graphical User Interface

Kernel -> Filter, Mask, or Convolution matrix

LGN -> Lateral geniculate nucleus

LIF -> Type of spiking neuron used in spiking neuron network models. The Leaky-Integrate-and-Fire neuron simulates one human nerve cell.

LIP -> Lateral intraparietal cortex

PFC -> The prefrontal cortex

Postsynaptic neuron -> The neuron receiving spikes from (presynaptic) neuron vis synapse that connects them.

Presynaptic neuron -> The neuron passing data via synapse to another neuron called postsynaptic.

RGB -> Red Green Blue

SNN -> Spiking Neural Network

Spike -> Action potential

Spike train -> The sequence of spikes that a neuron generates for a given time.

STDP -> Spike-timing-dependent plasticity

Unit circle -> A circle with radius 1.

V1 -> Primary visual cortex

V2 -> Prestriate cortex

WTA -> The winner-take-all method

V5, MT -> Middle temporal visual area

V6, DM -> Dorsomedial area

Chapter 1

Introduction

The introduction chapter will present the scope of the project, its structure and the Covid-19 impact on the development.

1.1. Motivation

In the modern world, the visual attention technique is applied to many applications in order to understand the content of a provided visual data. The technique enables users to identify specific objects and violations. For example, exist a security cameras program that implies the visual attention technique to detect weapons [1]. This application alerts the police or the security for a possible dangerous situation are improves human safety.

Due to its various detecting opportunities, the technique burst into popularity. Many companies started producing much more complicated algorithms and models deploying the main idea of visual attention. For instance, NVIDIA [2] and Tesla [3] invested in developing complex visual attention systems for inventing the vehicle of the future (autonomous vehicle). The self-driving teams of these companies became responsible for creating tools for detecting many different objects surrounding the car while moving and creating a safe environment for the passengers. Example of such methods is the NVIDIAs DNNs for detecting and classifying signs and traffic lights [4]. This enables the AV to recognize when it needs to reduce its speed and when it should stop to prevent a potentially dangerous situation.

The visual attention technique can be implemented in many different ways. The most common variations include the use of convolutional neural networks and the use of reinforcement learning. However, in the last few years, the SNN models became preferable due to their ability to mimic the activity of the visual system in the human brain. This SNN method is even considered much more computationally powerful than the other two. An example of a SNN model is the visual attention and object naming in humanoid robots. The model empowers a robot to recognize and name a particular object [5].

1.2. Objectives

The major aim of the project is to evaluate the use of brain-inspired SNN in the visual attention application.

In addition, a more detailed objectives plan is created:

1.2.1. Learning and exploring

- To learn about visual attention areas of the human brain
- To understand the spiking neural network components and methods
- To discover image processing techniques

1.2.2. Development

- Creating a method for transforming an image into spike trains
- Building a spiking neural network for visual attention
- Developing a GUI tool allowing users to interact with the SNN network
- Exploring how different parameters affect image spike trains
- Experimenting with different road images and analyzing results from the developed network
- Evaluating the visual attention model

1.3. Report Structure

The report is composed of 7 main chapters:

- Chapter 1 -> Introduce the reader to the topic of the dissertation and presents its aims.
- Chapter 2 -> Provides background for the neuroscience processes and describes image processing and deep learning techniques.
- Chapter 3 -> Presents the design of the developed visual attention system and provides all methods included in the network.
- Chapter 4 -> Discusses the implementation of the spiking neural network model and the technology applied in the development.
- Chapter 5 -> Analyzes experiments and evaluates the performance of both major parts of the developed application.
- Chapter 6 -> Presents ideas for future development.
- Chapter 7 -> Gives a summarized view of the personal achievements.

1.4. Impact of Covid-19

The Covid-19 pandemic disabled the normal student life and turned the university into online education. This caused constant stress due to the unavailability to meet other students and use university facilities and resources.

In addition, all of my traveling attempts to Manchester resulted in canceled flights due to the government travel restrictions. This reduced my motivation and left me with limited resources. While unable to return to campus, I and my brother had to share the same working space and technology which reduced my productivity.

Chapter 2

Background

The chapter provides the reader with the background knowledge required for understanding the following chapters. In the beginning, the visual attention concept is described. Then the most crucial areas and concepts of the neuroscience field are explained by focusing on the processes happening in the human visual system. Secondly, major image processing techniques are presented to show how visual input can be modified. In the end, the spiking neural network model will be discussed, as well as the helpful functions it provides.

2.1. Visual attention

Visual attention is a principle of focusing and orienting a specific object or task from a given visual input [6]. This principle removes the surrounding noise of the image by applying image processing techniques and follows the specified object while it is moving in the inputted video. Its purpose is to replicate the functionality of the human visual system.

2.2. Neuroscience and human visual system

Neuroscience is a biological field studying the nervous system and its impact on human behavior [7]. The field discovers many crucial processes happening in the human body. Examples include brain activity and the relation between the nervous system and the emotions of a human.

The human visual system is one of the most observed areas in neuroscience. It is placed in the human brain and comprises several areas responsible for understanding information received from the human eyes. Structurally, the system is divided into around 10 areas. The major groups of these areas are the retina, the lateral geniculate nucleus and the visual cortex.

2.2.1. The neuron, action potential and neurotransmitters

Humans brain contains around 86 billion neurons transmitting data between each other with synapses by generating spikes [8]. Different groups of neurons have different purposes. Few neurons are responsible for recognizing the visual data, others for the sound data and the remaining for all different tasks and processes in the human body.

The neuron is a composition of a body cell, dendrites and an axon that stores and passes information (Fig 1.). The dendrite acts as a receiver of the information (spikes). It then consequently passes the acquired data to the body cell and the axon. The axon creates action potentials (spikes) that are sent to the other neurons via synapses. When action potential enters the synapse, it releases different chemicals (neurotransmitters) that are passed to the postsynaptic neurons. When neurotransmitters are received they can pass the spikes from the presynaptic neuron (excite) or they may not allow spikes to enter the neuron (inhibit). All inhibitory and excitatory inputs contribute to the creation of an action potential threshold that determines whether and how many spikes will be emitted (fired) from the presynaptic neuron [9].

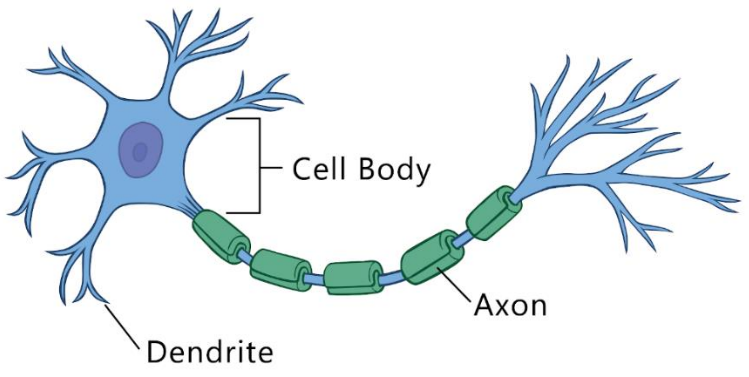


Figure 1: A neuron and its components, by Doctor Hanson [10].

2.2.2. The retina and the lateral geniculate nucleus

The eye is the first place, where the human interacts with the light and the surrounding background. The organ enables the human to detect different visual inputs it receives and translate them into spikes that can be interpreted from the neurons it is connected to. The whole process is performed from the retina part of the eye that is located in the inside of the eye.

Following the retina, the data is processed by the lateral geniculate nucleus. The LGN then determines the 3D dimensions of the images (spatial dimensions) and transports the updated data to the primary visual cortex (V1).

2.2.3. Visual Cortex and Lateral Intraparietal Cortex

The visual cortex is part of the brain that is responsible for identifying the basic characteristics of the visual input. The area can identify the color, orientation and shape of the visual data. The visual cortex is composed of 6 major parts that facilitate this functionality: V1, V2, V3, V4, V5 and V6 (Fig.2).

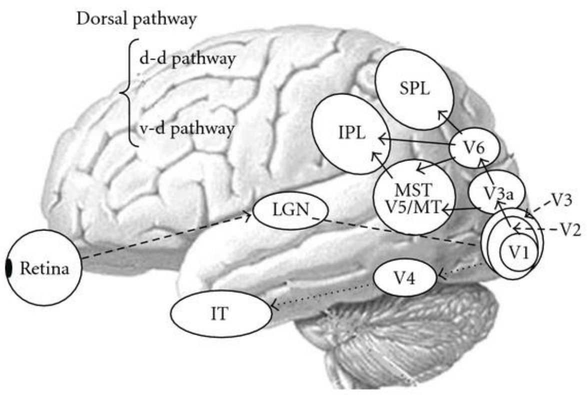


Figure 2: Image of the brain and the parts of the visual cortex, by ResearchGate [11].

The V1 area is performing the first processing of the data. It is tuned for color and orientation and identifies primitive aspects of the information like small edges or changes in the color [12]. The V1 area passes the acquired information to the V2 one. V2 combines partly the components of the information it is receiving from V1 and forms better-looking colors and edges. Like(wise) V1, V2 is tuned for color and orientation. Both V1 and V2 sends data to V3 and V4. V3 performs color selectivity, while V4 starts recognizing small shapes and is tuned for objects. The V3 area passes information to V5 and V6. Both V5 and V6 are tuned for movement and motion.

In addition, the V4 area transfers data to the Lateral Intraparietal cortex (LIP). The LIP area of the brain stores data for the location of the object that humans are looking at.

2.3. Image processing

The method for modifying a picture by blurring it or performing a change function on it is called image processing. Image processing enables the transformation of an image in a way to detect important features from it. Examples of such features may include edges, specific objects, or background.

2.3.1. The image

Before discussing the methods image processing performs, the structure of the image is explained. People use images to store visual data in files. They can be created with a camera or other technology that captures pictures. Images are the closest tool resembling the human eye that people have created. They can be RGB, grayscale or stored as another type.

The image consists of many pixels. In the RGB images, the color of every pixel is a triple (r, g, b) defined by how saturated are all of the 3 main colors: red, green and blue (r, g and b can store values between 0 and 255). In the grayscale ones, a pixel can have a value between 0 and 255 showing what nuance of the dark color it is storing.

In image processing, the grayscale images are preferred, because they are measured from 1 color dimension, while the RGB images need to process all of its 3 color dimensions in each method. For this reason, every image is converted into a grayscale one, before applying any process to it. The method of converting a RGB image to a grayscale image can be seen in Fig. 3.

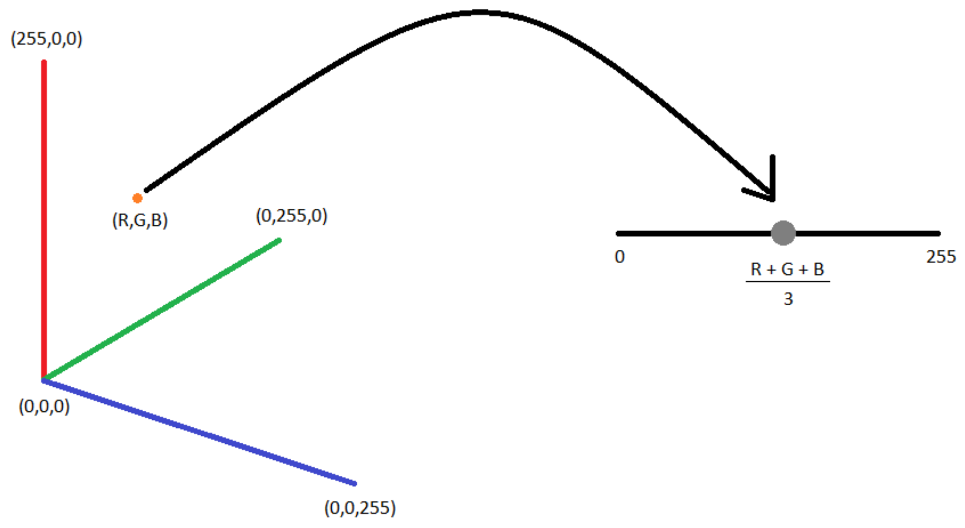
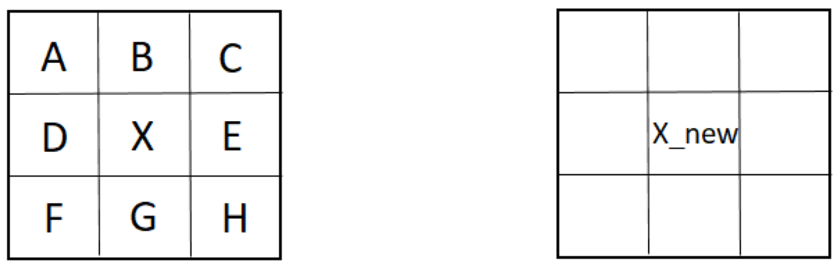


Figure 3: The figure visualizes the process of converting a RGB pixel to a grayscale pixel (it uses the average value).

2.3.2. Local normalization

One of the crucial functions of image processing is local normalization. Local normalization is the process of changing the intensities of every pixel with the mean value of its initial intensity and the intensity values of its neighboring pixels. This process updates the intensities in a way to reduce errors caused by shades, noise, illumination and other uneven shapes affecting the main object (ex. tree branch placed between the street and the camera when detecting the road). The conversion process for a pixel and its 8 surrounded pixels can be observed in Fig.4.

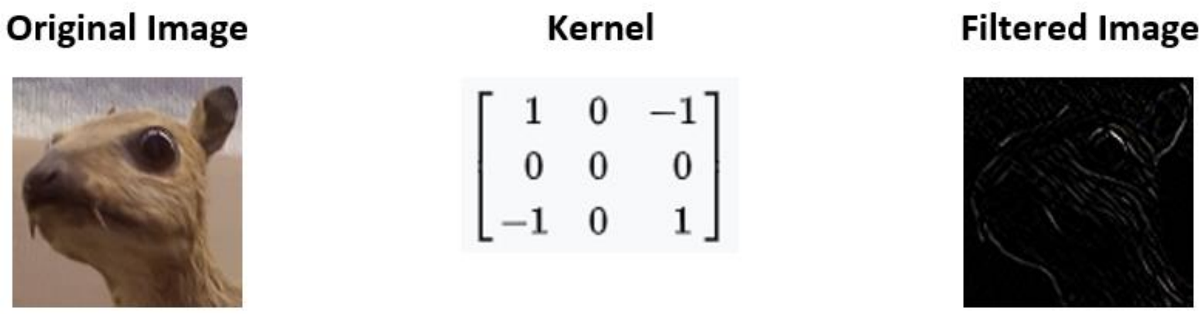


$$X\_new = \frac{A + B + C + D + E + F + G + H + X}{9}$$

Figure 4: The figure displays the function that updates the value of pixel X.

2.3.3. Kernels

Another important concept in image processing is the kernel. The kernel is a two-dimensional matrix that is applied to a picture in order to detect edges and important objects. For example, let’s assume that a street image contains a road, trees and 3 people. In order to detect the street, a specific kernel is applied to remove all unnecessary objects such as trees and people from the image. Another example can be observed in Fig. 5.



$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Figure 5: Example of a kernel applied to an animal image to detect its body edges, by Nuruzzaman Faruqui [13].

2.3.4. Gaussian filters

Another version of the kernel is the Gaussian filter. Its purpose is to detect edges in an image. The value of each cell in this kernel is the result of a Gaussian function applied to the position of the cell in the kernel. Usually, for an image, a 2-dimensional Gaussian function is applied, because the image has 2 dimensions. The Gaussian formula and the 2-dimensional one can be found in Fig. 6a. The distribution of the function returns high values for a small number of input values and values just over 0 to all other inputs. This property is very useful for reducing noise and focusing on edges and other crucial areas.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(m-x)^2}{2\sigma^2}}$$

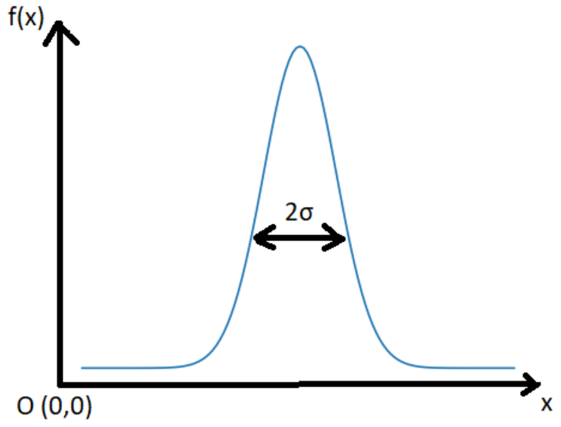


Figure 6: The top equation shows the Gaussian formula. In the equation σ is the variance of the distribution and m is the mean of the distribution. The figure under the equation presents the Gaussian distribution.

2.4. Spiking neural networks

One of the tools used to create a visual attention system is the spiking neural network. Spiking neural network is a method created to mimic the activity of the human nervous system. Every SNN is composed of several layers presenting different groups of nerve cells such as V1 and LIP areas in the brain. Each layer comprises many neurons replicating the activity of the human neurons by transmitting spikes only if a specific firing threshold is reached. Spike data is passed through the input layers. Then the information is processed by a number of hidden layers that transform the data. In the end, spike trains are returned from the output layers. Neurons between two layers are connected with weighted connections simulating synapses. Moreover, a few of the layers play a crucial role in the network. Examples of such layers are the convolution, the pooling and the competition layers. These special layers reduce and choose the fired neurons in a similar way to our nervous system. This enables the recognition and detection of particular features. Fig. 7 presents a simple spiking neural network composed of an input layer that has n neurons and an output layer with 1 neuron.

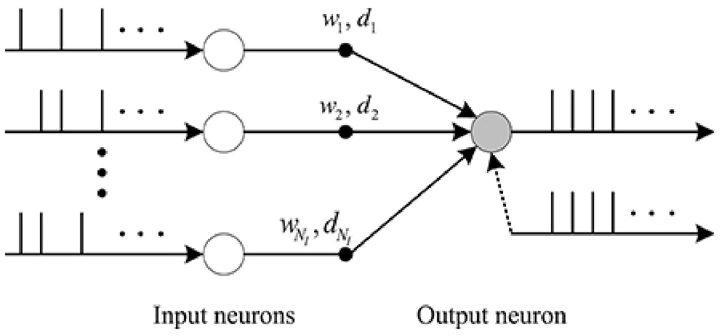


Figure 7: A SNN model having two layers: input layer and output layer. The input layer has N neurons that receive spike data. The output layer has only one neuron that returns the resulting spike trains [16].

2.4.1. Spike-Timing Dependent Plasticity

The most crucial process in spiking neural networks is the spike-timing dependent plasticity. The STDP process changes the synapse strength between two neurons based on the action potential times of the presynaptic and postsynaptic neurons. If the presynaptic sends spikes frequently before the postsynaptic neuron produces an action potential, the connection is enhanced. On the other hand, if the postsynaptic neuron occasionally spikes before the presynaptic neuron, the synapse connection becomes weaker. The weights of the connection are updated with the result of the STDP modification function shown in Fig. 8.

$$f(\Delta t) = \begin{cases} A_+ e^{(\frac{\Delta t}{\tau_+})}, & if\ \Delta t < 0 \\ A_- e^{(\frac{\Delta t}{\tau_-})}, & if\ \Delta t > 0 \end{cases}$$

Figure 8: The equation of the STDP is presented. A_+ and A_- display the amplitudes determining the synaptic modification, the variable t is denoting time and r_- and r_+ define the range of the appropriate spike interval [14].

2.4.2. Convolution layer

One of the key components of the spiking neural network is the convolution layer. It is used to extract specific features from a given data. When performed on a 2-dimensional input such as an image, the sizes on the data are reduced and the desired feature is extracted by using a filter on the original data. The size of the new data depends on the kernel size, the padding and the strides used on the original image or data. Padding is the rows of empty data added to the original data to protect crucial features (in a 2-dimensional image zero pixels can be added in all of the 4 edges to preserve edge features) or to enable the kernel to fit in the data. Strides present the number of pixels or data cells that are skipped from the filter. For example, part of the convolution process when strides are used can be observed in Fig. 9.

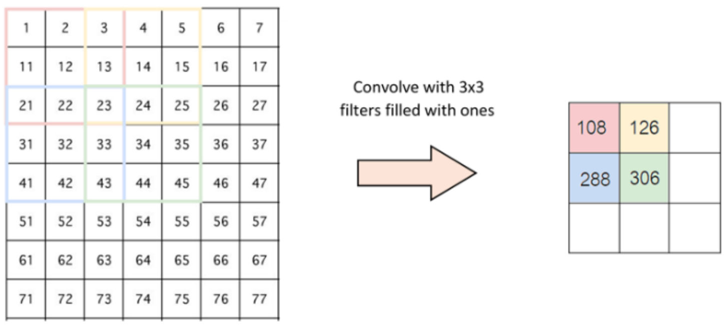


Figure 9: Example of a convolution process with a kernel of size 3 and stride 2 [27].

2.4.3. Pooling (subsampling) layer

Sometimes, the input data is large and needs many neurons to process it. This may reduce the computation especially if the network contains many layers. A solution for this issue is using a pooling layer. The pooling layer decreases the data by replacing parts of the data with a value received when kernel (or function for the values in the selected area) is applied to the area. The most common method is replacing the area with its average, maximum or minimum value. An example of maximum pooling is displayed in Fig. 10. The pooling layer enables the formation of more recognizable features and reduces the noise in the data.

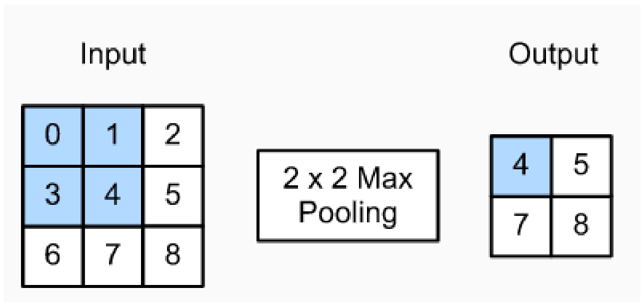


Figure 10: An example of max pooling process with a kernel of size 2 [15].

2.4.4. Winner-take-all technique

Another crucial method that the spiking neural networks are applying is the winner-take-all technique. The winner-take-all method creates a competition between layers that compete for selection. Two different versions of the principle are available. The first version is called a hard WTA and chooses only 1 neuron that has the highest spike value. The second one is the soft WTA, which enables several neurons to spike. In the soft WTA a specific function is applied to all neurons and acts as a threshold. The principle is very effective when the correct color or orientation is selected.

Chapter 3

Methodologies

Chapter 3 introduces methods created in the developed application. It presents formulas for different Gaussian filters applied to images and an approximate visualization of the corresponding kernels.

3.1. Areal filtering

Before presenting the formulas used to evaluate the values in the Gaussian filter, an understanding of the Areal filtering is required.

Areal filtering is a 2-dimensional Gaussian filter for separating and differentiating between different surfaces and uneven areas (with unequal wavelengths) [17]. It contains the same parameters as the Gaussian filter. However, it does have x and y parameters instead of only a x parameter. In the created SNN network, 5 different Areal functions are implemented. Each of the functions corresponds to one of the five directions: horizontal, vertical, diagonal, anti-diagonal and circular (multiple dimensions). Four of these five functions are dependent on the angle θ. θ is the angle in the unit circle between the x-axis and the line going from the origin to point A on the unit circle. Point A is located in the first or the second quadrant of the unit circle and is related to the used orientation (Fig. 13). This angle stores a value of 0 when the orientation is horizontal, 45 when it is diagonal, 90 when it is horizontal and 135 when it is anti-diagonal.

The functions for the horizontal and vertical follow a similar pattern P, as well as the functions for the diagonal and anti-diagonal orientations which follow pattern P2. The idea behind both patterns is to create a line in the kernel emphasizing the orientation while reducing the effect of the other kernel cells depending on their line distance. The first pattern P can be displayed with the formula $(xcos\theta)^2 + (ysin\theta)^2$and the second one P2 with the formula $(xcos\theta - ysin\theta)^2 + (ycos\theta - xsin\theta)^2$, where x and y are the coordinate positions of the cell in the kernel.

The multi-dimension characterized as a circle was inspired by the famous formula $x^2 + y^2 = 1$ which depicts all points on the unit circle. The core idea is to simulate a circle in a kernel. This is achieved by giving higher value to the cells which are closer to the perfect inscribed circle in the kernel and lower values to the cells further from the circle points. The implemented formula is $r^2 - x^2 - y^2$, where x and y are the positions of the cell in the kernel and r is the kernel size divided by 2 (the radius of the circle). It does not create a real circle, because a circle cannot be created only from squares. However, the method builds an outstanding estimation of a circle filter. The full formula and the kernel are presented in Fig. 11.

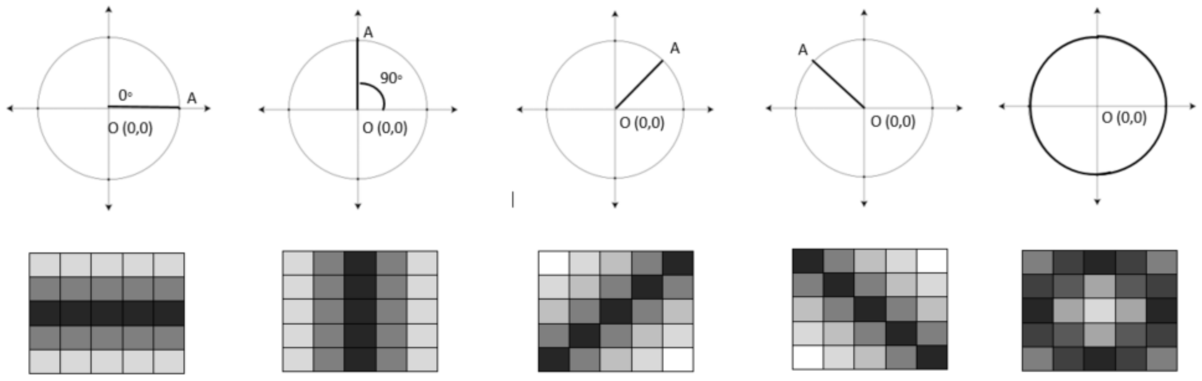


Figure 11: The correspondence between the angle θ and the kernel. The top row of the figure presents where point A is placed on the unit circle. The bottom row visualizes the approximate kernels the corresponding function is producing.

3.2. Cognitive architecture of the model

The visual attention comprises two components (Fig. 12). The first one is the process of creating spike trains from a visual input. The second component is the spiking neural network that processes the spike trains and performs orientation selection.

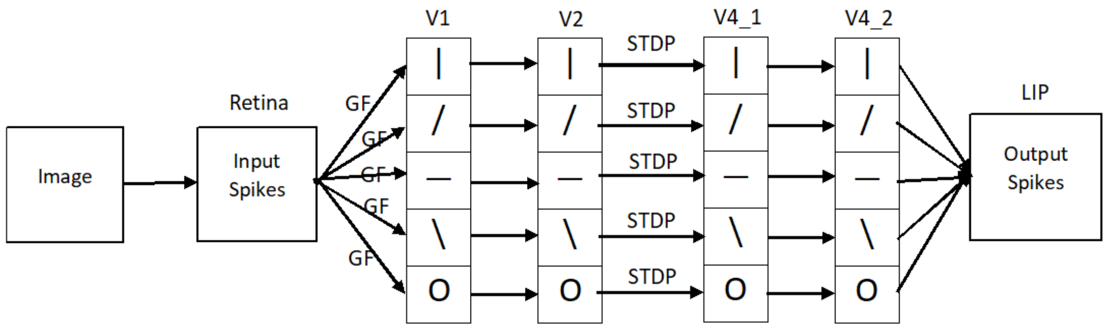


Figure 12: The design of the visual attention network.

3.2.1. The process of creating retina like spike trains

The main idea of the process is to create spike trains from the intensity of the pixels. Firstly, the visual input is converted into a square sized image (16 * 16 pixels) in order to create symmetry. Then a local normalization is applied to reduce noise and emphasize the main features. Finally, the values are transformed to spike trains via intensity to latency transformation. In the intensity to latency transformation, the possible pixel values (256) are divided with the time provided for spiking (the runtime of the program) to receive a “time step”. This “time step” is then applied as a measure that states from when a specific pixel value is spiking. Pixels with higher values spike sooner than pixels with lower values. For example, if an image containing 2 pixels with values A and B (A < B), the pixel with value B will spike first and will produce a larger number of spikes than the other pixel.

3.2.2. Population

Before presenting the structure of the spiking neural network, the term population is defined. The population is the entity of every area of neurons in the neural network. It acts as a data structure for storing vital data. Each population has a square shape and looks like a matrix. For example, the retina is population, while huge areas such as V1 in the network is comprised of 5 populations.

3.2.3. Cognitive architecture of the spiking neural network

The SNN design of the visual attention model is inspired by the network described in the Integrated Neuromorphic Visual Attention network [18]. In similar way parts of the Retina, V1, V2, V4 and LIP layers were created. However, the PFC layer is considered redundant in the network, because not all objects have preferred orientation. For example, streets can be oriented in many different orientations depending on their type and purpose. Unlike objects having parallelepiped forms or objects with non-standard forms such as cups and boxes, streets do not have volume. In addition, the model included an additional layer V4_2 for better recognition. A visual representation of the structure is displayed in Fig. 13.

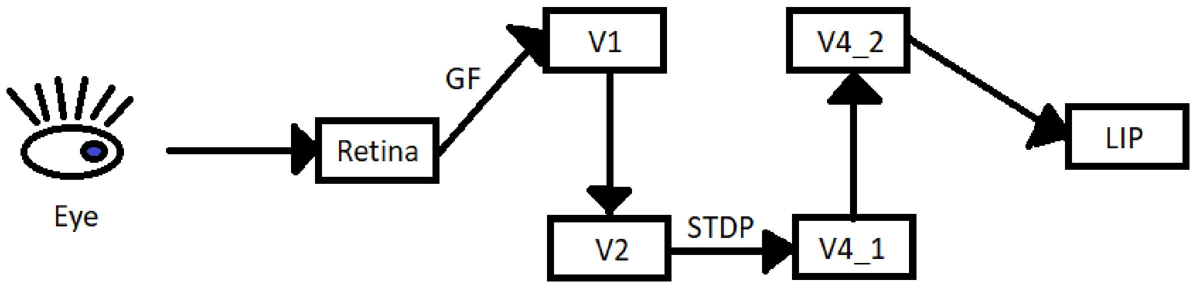


Figure 13: The design of the spiking neural network.

In the network, every layer has a specific role and tries to replicate a different part of the visual system in the human brain as its name is stating.

The Retina layer is a population with a size 16 * 16 neurons. It stores the visual information in a form of spikes. It acts as an input layer just like the human retina.

Layers V1, V2, V4_1 and V4_2 from the second part of the network are composed of LIF spiking neurons and mimic the behavior of the visual cortex. All of these layers are divided into 5 population areas. Each area corresponds to one of the 5 orientations: horizontal (--), diagonal (/), vertical (|), anti-diagonal (\ ) and multi-dimensions/ circle (O).

The V1 layer act as a convolutional layer. All five populations of the V1 layer have 12 * 12 neurons that act as the first layer that performs an orientation selectivity. The retina layer connects to each V1 area with a different areal 5 * 5 filter (explained in the first part of the chapter) depending on the orientation of the connected population. When the information is received, each population in V1 sends data via a single connection to the corresponding neuron in layer V2 in the area of the corresponding orientation. In a similar manner to layer V1, layer V2 has 12* 12 neurons in each of its populations and performs orientation selectivity on a higher level.

Layers V4_1 and V4_2 group different features such as edges and curves into objects. The areas of both layers contain 6 * 6 neurons. The layers simulate the activity of two of the components of the V4 area in the human brain. The V4_1 layer is a pooling layer that subsamples the data with 25% per orientation. This is achieved by performing an average pooling on non-overlapping 2 x 2 areas (kernel of size 2 x 2 is applied and a stride of 2). The V2 uses a Spike-Timing Dependent Plasticity when connecting each neuron with the corresponding neuron in all populations in V4. Afterwards, neurons in V4_1 are connected with a one-to-one connector (connected with their corresponding neuron) to the neurons in V4_2.

In the end, the LIP layer selects the orientation with the strongest action potentials. The layer is a population of 6 * 6 LIF neurons and acts as a decision layer. The neurons in each population in the V4_2 layer are all connected to their correspondent in the LIP population. Finally, a hard winner-take-all technique is performed to select the visual attention results which identify the position of the object.

Chapter 4
Implementation

The chapter presents the structure of the developed network and the relations between different components. It states the technology applied in the development process and the datasets used for the experiments in the following chapter.

4.1. Technology

The network was written entirely in Python3.8. The programming language was chosen because it is providing a huge number of already developed deep learning libraries. Several examples include TensorFlow [19], Keras [20] and PyTorch [21].

In addition to python3.8, many libraries have been included in the model. The major one is Numpy. Numpy is a famous useful python tool for storing and manipulating data stored in an n-dimensional array. In the model, it is used alongside the Tensor as a tool for storing image information. Also, math and mathplotlib.pyplot libraries were used for performing mathematical operations and visualizing data. Math is a fundamental python tool for performing all mathematical operations (ex. log, sin, cos). It plays a key role in the calculation of the Gaussian filters. Mathplotlib.pyplot is a tool for displaying data in graphics and charts. It enables users to analyze easier given data information.

Moreover, several libraries were applied for the creation of the SNN network: Pillow, PyNN [22], NEST [23] and SpikeTorch [24]. The Pillow library is allowing images to be loaded, stored and manipulated (ex. applying filters on images). PyNN [22] is a python tool for creating and building neural networks. It contains several methods for creating populations and connections between layers. This tool has a major role in the project as it is used to build the spiking neural network. NEST [23] is one of the three major simulators (NEURON, NEST and Brian) developed for running the PyNN code. SpykeTorch [24] is a version of PyTorch [21] that includes the concept of time. This recently created library enables the user to create spiking neural networks. In the network, the process of converting a picture into spikes is performed with SpykeTorch. The SpykeTourch Tensor object is a data structure for storing data in n-dimensional form (ex. a matrix). The information it stores can be manipulated but two different types of data are not allowed to be stored in a Tensor.

Finally, Tkinter was chosen to create the GUI application.

4.2. Datasets

Two different datasets are used for the experiments performed on the created application. Both of the datasets provide access to numerous road images. The first one is the Unsplash [25] dataset. This dataset is the biggest online open dataset for images. The data consists of images of real-world objects taken from many photographers across the world. The second dataset is one of the road dataset the Michigan Institute of Technology has built. The data is part of the CBCL StreetScenes Challenge Framework [26] developed from their representative and contains many real-world images of streets.

4.3. The implementation of the visual attention model

The implementation is composed of three main components: a class for converting a visual input into spikes, a tool for creating the spiking neural network described in chapter 3, and a visualization part for displaying different aspects of the model in a more user-friendly way. A diagram of the first two parts can be observed in Fig. 12.

4.3.1. Converting images to spike trains

Before converting an image to spikes, a number of operations are performed on the image to produce the correct amount of pixels/ neurons emitting the correct spikes. In the beginning, the input image is resized to 16 * 16 (the number of pixels corresponds to the desired number of input neurons for the SNN). Then the resulted image is converted to grayscale in order to reduce the information that each pixel is storing. This operation reduces the time needed for creating spikes and for differentiating two different images. Secondly, a Tensor object is created for storing the brightness values of each picture. A new dimension is included in the newly created Tensor to store the spike information for each second (time dimension). Thirdly, a local normalization is applied to all pixels with the values of their 8 surrounded ones (3 or 5 if they are part of the corners/edges). This reduces errors caused by shades and illumination and other uneven shapes affecting the main object. Afterwards, an intensity to latency transformation is performed to convert the brightness values into spikes. This operation is performed with the SpykeTorch operation Intensity2Latency for a given time period (runtime of 100 seconds is producing good results). In the end, the updated time tensor (runtime, 16, 16) is reshaped into the array [N_1, N_2,…], where N_i is an array presenting the spike history of neuron i. This transformation enables the program to create valid inputs for the visual attention spiking neural network.

In addition, the spike method was updated to enable users to specify the desired image size (the dimension of the input population) and runtime. This method required a restriction to the minimum value the input values can store and the connection between them.

4.3.2. Tools for creating the spiking neural network

The second main component of the model is the spiking neural network that performs visual attention on the input spikes (described in chapter 3).

In order to create the network, a class has been built for creating layers. The class enables users to create and update a layer consisting of N populations (N is a number provided by the user). It also provides methods simulating different types of synapses and operations. The functionalities include an all-to-all connector, one-to-one connector and the operations: Gaussian filters, subsampling and winner-take-all technique. Most of the connection types can include STDP.

In addition, a Network class was implemented for creating the spiking neural network described in chapter 3. The class creates 2 layers consisting of 1 population for the retina and LIP area and additional 4 layers consisting of 5 populations each for the V1, V2, V4_1 and V4_2 areas. Then the 5 Gaussian filters (stated in chapter 3) are applied to the retina input to produce the data in the V1 populations. Afterwards, a one-to-one connection with a static weight 7 connects the V1 and V2 areas. The V2 connects to V4_1 with static weight 12 and performs the subsampling technique. On this connection a STDP is also applied. Then a one-to-one connection is added between layers V4_1 and V4_2 with static weight 12. Also, connections between the V4_2 populations and the LIP population with static weight 9 are included. In the end, the winner-take-all method is applied on the LIP area with a static weight -12. Additionally, the class provides the opportunity to visualize the spike trains of every population in the network.

Moreover, the network class has been updated in a way to empowers the users to play with the model by changing its parameters. Users can specify the size of the input layer (the dimension of the input population), the orientations used, the size of the filter and the weights between layers before running the model.

4.3.3. Model visualization

In the end, a GUI was created to enable users to work with the program without the need for the terminal. The GUI allows the user to look for a specific component of the application in a more visually appealing environment. The built GUI is composed of 4 pages (Fig. 14).

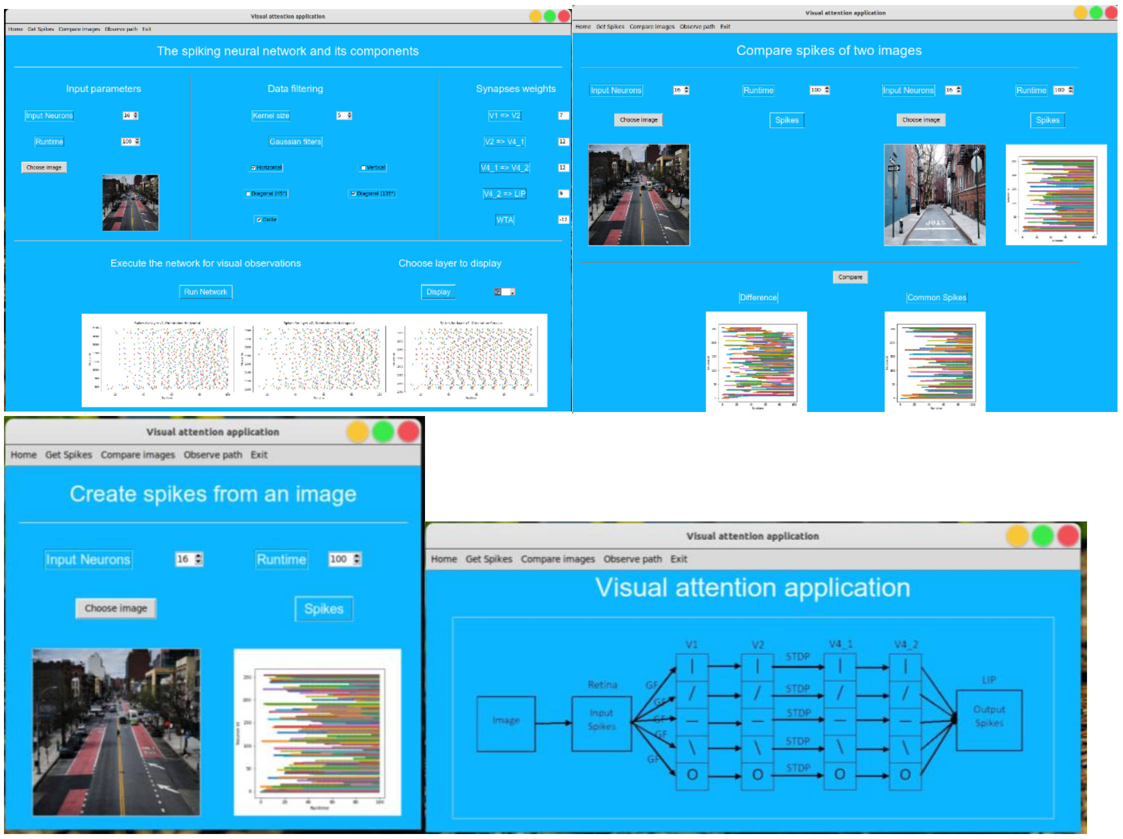


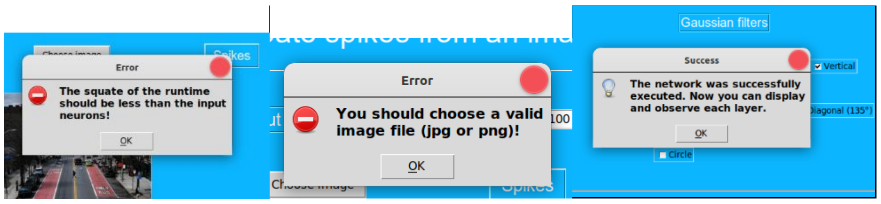
Figure 14: Recaps of different pages in the GUI.

The first page acts as a home page and displays the overall structure of the implemented network. The second one visualizes the method described in part 4.1. The user should specify the image, the size of the dimension of the input population and the runtime. After specifying this data, he can observe the time spikes for the given picture. The third one provides the user with the ability to experiment by comparing spike trains of two images. Before comparing them, the user needs to specify the image, the dimension size of the input population and the runtime for both pictures (this data can be different for the two pictures). After finalizing the process, the user can observe the time spikes for both images as well as their common spikes and the ones that appear only in one of the images.

The final component allows the user to observe how spike trains are progressing throughout the entire spiking neural network. The user again needs to specify the image, the dimension size of the input population and the runtime as well as the orientations he is willing to include in the SNN and the weights for each connection between layers. After the data is provided, the user can observe how spikes are changing in all layers of the SNN, except from the Retina one that can be observed in the first two components of the GUI.

Users are provided with an exit button if wanting to exit the program.

Moreover, error messages and tip boxes were implemented to inform and guide the user while he or she is using the application (Fig. 15). This prevents actions that may shut down the platform or pass unsupported type into one of the functions. The supported boxes give additional information that acts as a hint to the user. They may provide information for the source of the error, such as unsupported kernel size or a relation between parameters that the user does not know about.


Error

The squate of the runtime should be less than the input neurons!

Error

You should choose a valid image file (jpg or png)!

Success

The network was successfully executed. Now you can display and observe each layer.

Figure 15: Examples of errors and tips the GUI sends.

In addition, when a user changes any of the parameters, the application detects the change and removes all results it has displayed for the previous set of parameters. This technique ensures that the user does not receive a piece of incorrect information when parameters and results have changed. It also reduces memory and increases the speed of the program.

Chapter 5

Experiments and results

Chapter 5 presents few experiments performed on the created model. The chapter is divided into 2 main parts each experimenting with one of both major components of the network. Moreover, it analyzes results received from the network when different values of the parameters are applied.

5.1. Converting images to spikes experiments

In the first bunch of experiments, the process of creating spikes from different images is observed. Its purpose is to illustrate how spike trains change when the image size or the runtime is modified. Experiments are separated into 3 different sections. The first one is inspecting the difference between two spike trains when the input size is increased. The second one observes this difference when the runtime is rising. The third section compares spike trains received from images when the same set of parameters is applied to them. Two images of distinct types are used for all experiments. The first one is an image from the Unsplash dataset, while the second image is a cropped street with a white background (street image without the surrounding noise). Images can be seen in Fig. 16.

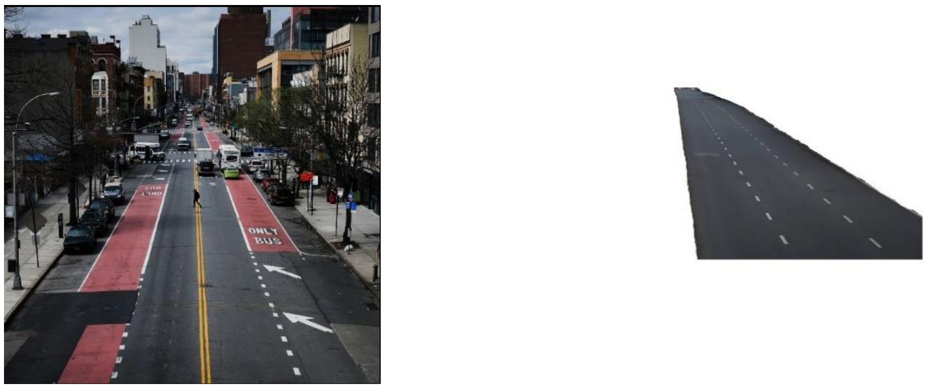


Figure 16: Images used for experimentation purposes.

5.1.1 Input size experiments

The experiment is conducted into two components. The first one is comparing an initially stated spike result to spike trains received from the model (described in chapter 3.4.1) when the identical image is inputted with larger input size. The second component is comparing spike results of an image with size N * N to the spike trains received when the identical image is inputted with a size (N - 1) * (N - 1).

5.1.1.1. Experiment comparing results to initial value

The experiment compares spike trains of an image with input size 7 and runtime 49 to spike trains received from the image with runtime 49 and input size N. The experiment is performed for values of N between 7 and 32 inclusively (7 <= N <= 32). A visual representation of the results can be found in Fig. 17.

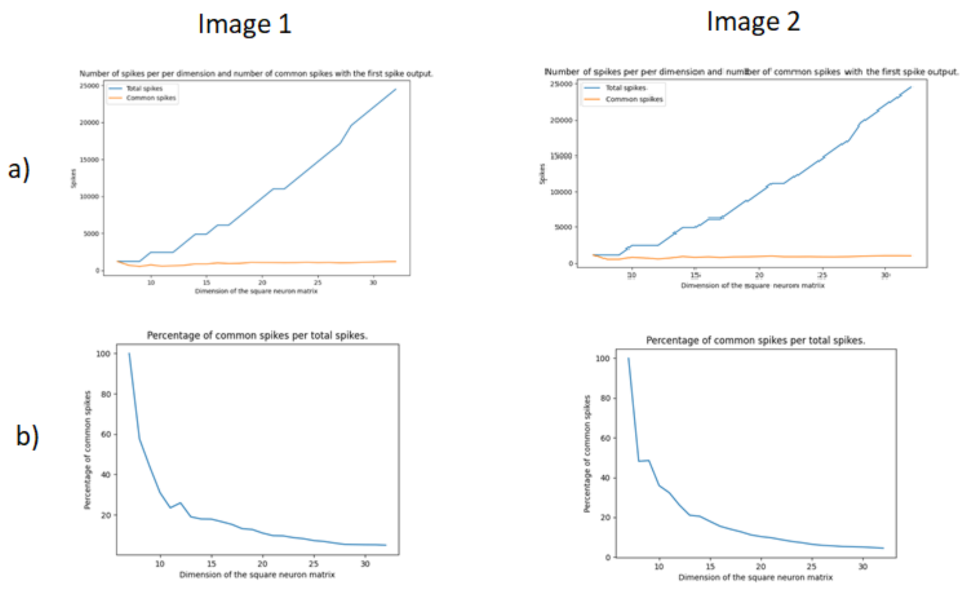


Figure 17: a) Shows the total spikes for input size between 8 and 32 and the common spikes with the spike trains received when the input size is 7. b) Presents the percentage of common spikes out of the total for given input size.

From the results, it can be observed that when the input size is increasing, the common spikes with the initial spike trains are decreasing. This phenomenon appears because when the image size is increased, new neurons are created. Logically, while increasing the new neurons, the total number of spikes rises. This results in a smaller percentage of spikes appearing in the first 7 * 7 neurons out of the total number of spikes.
Another interesting observation that the graphs represent is that the total number of spikes does not grow linearly when the input size increases. A possible cause could be the nature of spikes. When the size is increased, the new neurons may not produce any spikes, because represents white background (containing no noise).

5.1.1.2. Experiment comparing to the previous result

The experiment compares spike trains of an image with input size N and runtime 49 to spike trains received from the identical image with runtime 49 and input size N - 1. The values of N are similar to the values used in the previous experiment in order to provide the opportunity to compare both results. N takes values between 8 and 32 inclusively (7 < N <= 32). The results of the experiments are shown in Fig. 18.

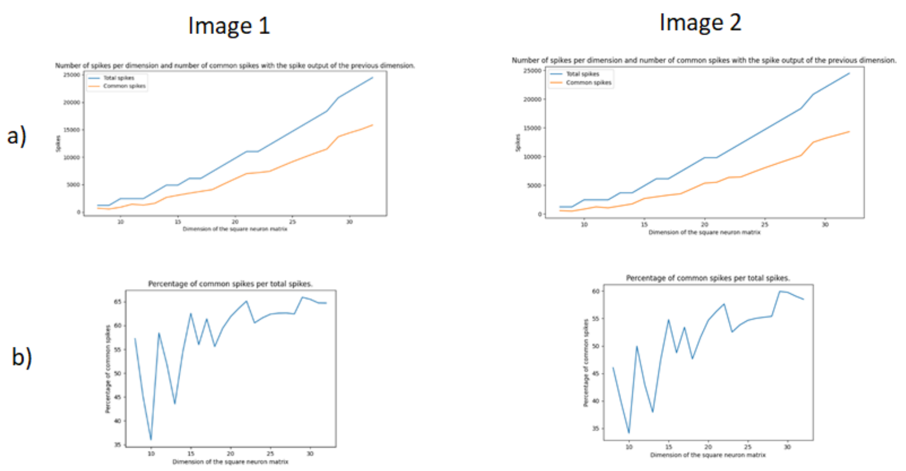


Figure 18: a) Shows the total spikes for input size between 8 and 32 and the common spikes with the spike trains received when the previous input size is used. b) Presents the percentage of common spikes out of the total for a given input size.

Just like in experiment 1, both images produce similar results. However, the percentage of common spikes for the first image has a bigger amplitude for values of N between 8 and 15 than the percentage of common spikes for image 2. A possible cause could be the surrounding noise of the image. For the results of both images, can be stated that the percentage of common spikes is increasing when the N rises. Additionally, the percentage results for image 1 show a higher similarity between spike trains of two consecutive input size values. The reason could be that image 1 has a surrounding noise that produces additional spikes, while when the input size increases in image 2, few white “pixels” (neurons presenting the data of a pixel) are changed to pixels that are part of the street. This produces spikes that are not in the previous spike trains (the one received from the smaller input size). Moreover, it can be observed than the percentage of common spikes is much higher that the percentage of spikes received from the previous experiment for all values N higher than 8.

5.1.2. Runtime experiments

In a similar way as the previous large experiment, this one is conducted into two components. The first component compares an initially stated spike result to spike trains received when the identical image is inputted with a larger runtime. The second one compares the outputted result of an image with a runtime R to results received from an image with a runtime R - 1.

5.1.2.1. Experiment comparing results to initial value

This experiment compares the spike trains of an image with input size 16 and runtime 10 to spike trains received from the image with input size 16 and runtime R. The experiment is conducted for values of R between 10 and 200 inclusively (10 <= R <= 200). Results can be observed in Fig. 19.

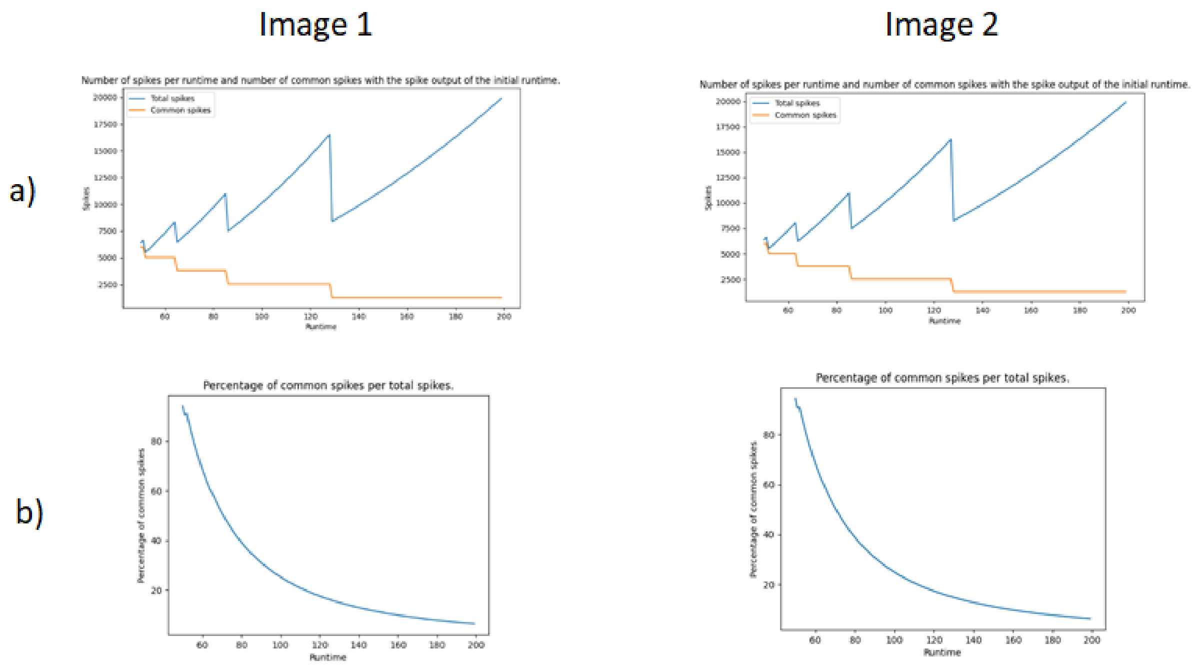


Figure 19: a) Shows the total spikes for runtime between 10 and 200 and the common spikes with the spike trains received when the runtime is 10. b) Presents the percentage of common spikes out of the total for given runtime.

The results of both images are almost identical. The number of common spikes is decreasing slightly (on stages), while the total number of spikes is fluctuating. A possible cause for this phenomenon could be that neurons are spiking much later when the runtime is increased in order to create a more realistic effect. In addition, the percentage of common spikes is not significantly declining except for the initial values of R. Again, this could be a result of spiking much later due to the larger time.

5.1.2.2. Experiment comparing to the previous result

The experiment compares spike trains of an image with input size 16 and runtime R to spike trains received from the identical image with input size 16 and runtime R - 1. R takes values between 50 and 200 inclusively (50 <= R <= 200). The results of the experiments are displayed in Fig. 20.

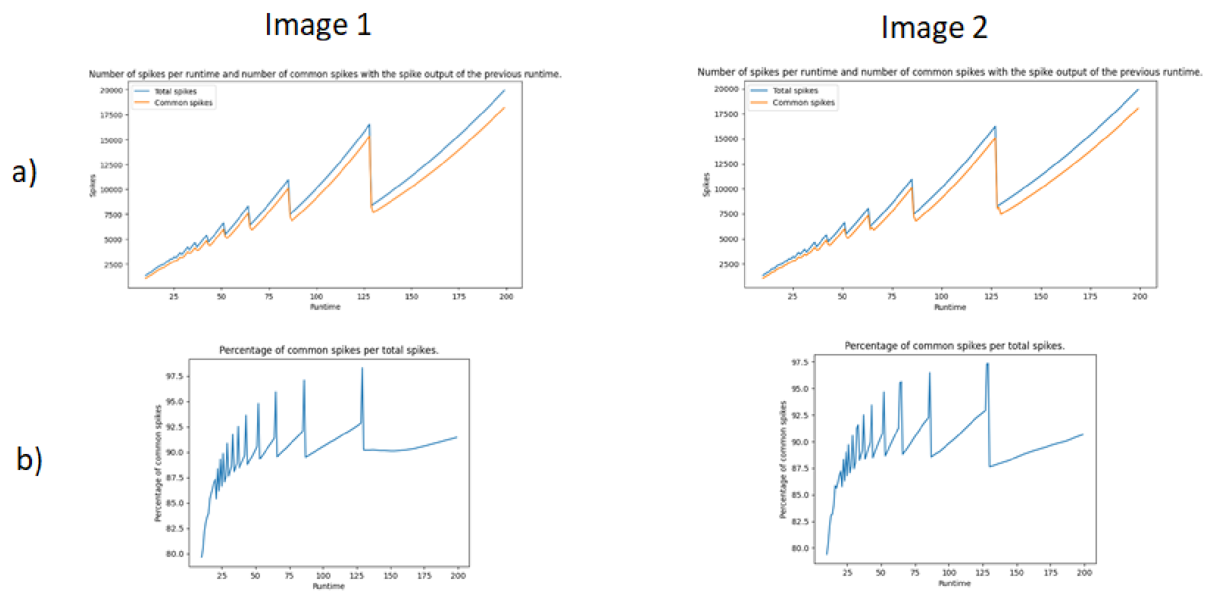


Figure 20: a) Shows the total spikes for runtime between 50 and 200 and the common spikes with the spike trains received when the previous runtime is applied. b) Presents the percentage of common spikes out of the total for given runtime.

Results for both images are identical. The percentage of common spikes is very high. It has an average value of around 90% and achieves a value of approximately 98% for runtime 130. This implies that spikes are very similar between spike trains received from an image with a runtime difference equal to 1.

5.1.3. Experiment comparing image spike trains

The purpose of the experiment is to compare spike trains of an image to the spike trains of many other real-world images when identical parameters are applied. For all images, the input size is set to 16 and the runtime is set to 200. The results of the experiment can be seen in Fig. 21.

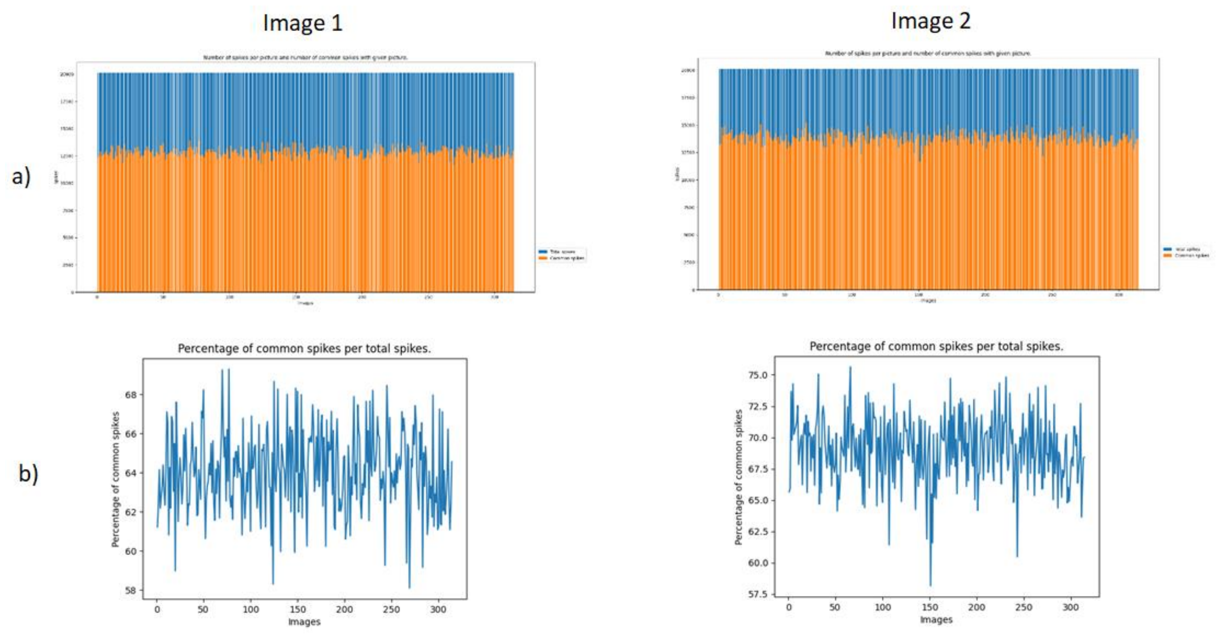


Figure 21: a) The total and common spikes of more than 300 images with both images from Fig. 16, when the input size is 16 and the runtime 200. b) The percentage of common spikes per image.

Results from the experiment show that both images have at least 57% common spikes with all of the compared images. The first image produces lower results compared to the second one. The main reason behind this observation is that the first image contains more noise (additional information) than the second one. An average of 70% is the percentage of the second image. This implies that around 70% of the content of the observed real-world images has similar properties as the second image (most of the real-world images have the same orientation).

5.2. Experiments on the spiking neural network

In the following experiments, the performance of the whole network is examined. Firstly, an experiment of Alex Rast's work is recreated to inspect the functionality of the network. Secondly, LIP results from images with approximately no noise are observed. In the end, an experiment presenting the spiking neurons in the LIP area for many real-world images is performed.

5.2.1. Replication of the original experiment

The experiment is a recreation of the experiment explained in the article Towards Real-World Neurorobotics: Integrated Neuromorphic Visual Attention [18]. Its goal is to achieve results similar to the results in Experiment I in the paper. The result of the paper can be seen in Fig. 22 a). There are 2 major differences between both experiments. The first one is that the LIP area in the paper is smaller. It consists of 25 neurons, while the current model has 36 neurons in the LIP area. The second difference is the time. In the original one, the runtime is 70, while in the current experiment it is set to 100. The same data was applied to the current network. It was generously provided by Alex Rast. Results of the current experiment can be observed in Fig. 22 b).

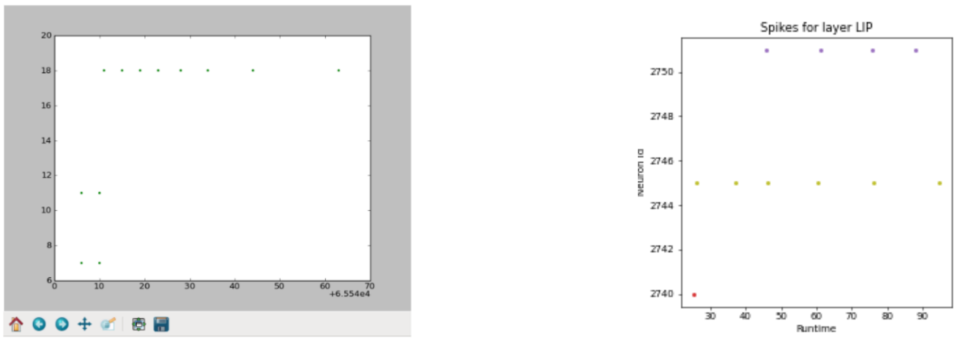


Figure 22: The spike trains of the LIP area for the experiment explained in the article Towards Real-World Neurorobotics: Integrated Neuromorphic Visual Attention [18]. The image on the left (a) shows the original results, the figure on the right (b) presents the results from the current visual attention model.

The experiment replicates very closely the result of the original network. It does contain only 3 neurons that are spiking like the original one. The result is not identical but it shows one neuron that is spiking frequently and two others that spike regularly or just once. Overall, the experiment is successful and outputs results very similar to the original one.

5.2.2. Experiments on images with almost no noise

This experiment evaluates the performance of the model with the spike trains generated from the spike conversion method. The experiment is conducted with two different images. Each of the images presents one of the main orientations: vertical and diagonal. Both images contain as little noise as possible. In the end, a result of the vertical image is observed when the input size is reduced. This will present how the output is changing when the input neurons are decreased. Images used in the experiment are present in Fig. 23.

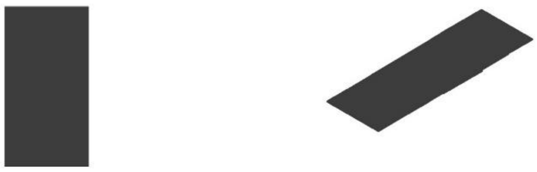


Figure 23: Street images hot containing surrounding noise such as trees.

For each image, a kernel of size 5 is used, and an input image of size 16 and a runtime 100. The LIP results for the vertical and diagonal images are visualized in Fig. 24 a, b.

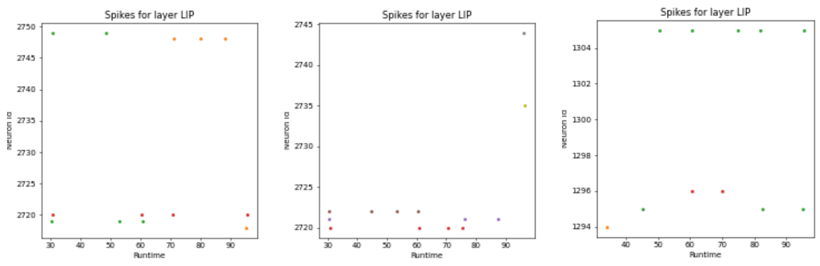


Figure 24: The first two figures (a, b) shows the LIP spikes for street images in Fig. 23 when the input size is 16 and the runtime 16. The right one (c) visualizes the spike trains of the vertical image when the input size is decreased to 12.

The result of the vertical image includes 5 spiking neurons that identify the position of the object. The output does include several neurons that spike because the width of the object is huge (only twice less than the length). However, most “active” neurons spike several times. Overall, the LIP output narrows the input to several neurons that may store the location of the top and the bottom of the vertical object.
The results of the diagonal image contain 5 neurons that produce spikes. If we observe Fig. 28 b) a slight noise can be observed on the diagonal edges. This may be the cause for the top neurons to appear spiking. However, the other 3 neurons that are placed closely (clustered) are spiking frequently. This presents that the visual attention is working and narrows the results to a single area of spiking neurons.

Finally, an experiment is performed on the vertical image with smaller input size. The input size has been decreased from 16 to 12, while the other parameters stayed identical. The LIP result of the network can be found in Fig. 24 c). The visualized output contains 4 spiking neurons placed in a similar pattern as the output of the vertical image with input size 16. This implies that weight modification is not needed when the input size is changed.

5.2.3. Experiments on real-world images

The last experiment evaluates the performance of the network on various real-world images. It calculates and displays the number and percentage of spiking neurons as well as the total number of spikes. The experiment is run on more than 300 different road images. Afterwards, the experiment is repeated for the same set of images but with a smaller input size. The input size of the images in the first experiment 16 while the input size in the second one is 12. Results can be observed in Fig. 25.

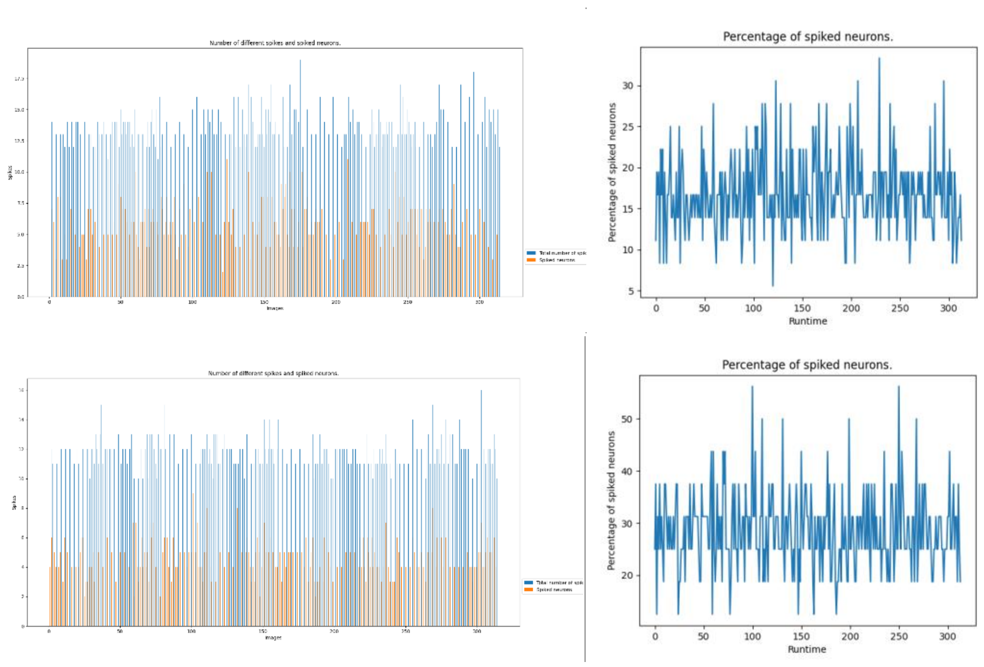


Figure 25: The left figures present the number of total spikes and the number of neurons that have spiked per picture. The right figures show the percentage of neurons that have spiked. A) The top figures present results from the original parameters of the visual attention network, while B) the bottom images present the results when an input size of 12 is applied.

The results from the first experiment show that the output of the network has an average of 15% neurons that are spiking. This gives around 5 neurons spiking for each image. Considering the noise each image contains, these results are very acceptable. In addition, it can be observed from Fig. 25 a) that most of these neurons are spiking at least 3 times for the given runtime of 100 seconds. This implies that inputs are narrowed to a small number of neurons.

The second experiment produces much more inaccurate results. The average number of spiking neurons is twice higher (30%). This produces around 5 spiking neurons for each image. However, the input size is reduced to 12 which transforms the LIP area into an area of 16 neurons. This implies that an average of almost half of the neurons are spiking.

Overall, the current parameters of the network produce considerably adequate results. However, the model requires additional modifications when smaller input sizes are applied in order to produce more acceptable results.

Chapter 6
Future development

This chapter displays possible improvements on the network and suggests features to be updated in future development. It proposes 3 different extensions that can be implemented in the future.

6.1. Replicating the brain’s visual system in greater detail

The network can be updated to mimic the human visual system even more closely. First of all, an additional spike input can be added. This will simulate the human second retina. Secondly, data from both retinas can be connected to a new area LGN with a one-to-one connector, mimicking the lateral geniculate nucleus in the human brain. Thirdly, the newly created LGN area can be connected to the primary visual cortex via synapses using the developed Gaussian filters. Then V3, V5 and V6 areas can be implemented to create a full representation of the human visual system. These three layers may enable the programmer to recognize not only orientations but also different shapes, colors and movements. Afterwards, new connections between different parts of the visual cortex can be applied to produce more accurate results.

6.2. Visual attention in video

Another idea for future development is allowing the network to operate on video segments. This will be beneficial because it will provide the opportunity to recognize objects while they are moving. For instance, detecting streets while the self-driving car is moving can be achieved. This may enable us to protect the AV from crossing road borders.

6.3. Recognizing the surrounding objects

In most cases, when the network is detecting a specific object A, it cannot recognize any other object types that are placed around object A. This can be achieved by changing the weights of the network and applying the resulted network with the original one. This might be a very crucial implementation because often two objects depend on each other. For example, when driving the autonomous vehicle should identify both signs and streets in order not to break any traffic laws.

Chapter 7

Reflection and conclusion

The final chapter presents a summary of the personal achievements. It discusses challenges that were faced throughout the project and the skills acquired while developing the application.

7.1. Challenges

When I started working on the project, I had almost no experience with visual attention systems. However, I was motivated to create and evaluate a visual attention and naming model mimicking the brain visual system entirely. At the time, I did not realize how complicated and ambitious goal I was pursuing. When I start researching the field of neuroscience and all components of the real visual system, I understood why recreating the whole system is almost impossible and why hardly exist available articles supporting the full recreation. Then, I reformulated my objectives and tried to build a much smaller example of the human visual system.

In addition, I was not familiar with any neural networks when I start developing my project. Most of the programming libraries supporting the DNN functionality contain a limited number of tutorials and example programs. This increased the difficulty of the task. I also needed to get familiar with the CNN library PyTorch to fully understand the updated version (SpykeTorch) which took additional time.
Moreover, searching for street datasets was very difficult. Most of the found datasets contained irrelevant information such as images taken of the sky or of a house.
Also, selecting the correct weights for the network was challenging. The main reason for this was that searching for the correct connections and Gaussian filters required a lot of time. However, after adjusting them I was delighted with the results.

7.2. Knowledge development

The project has been an excellent opportunity to develop and improve my knowledge in different fields. To begin with, I enriched my basic understanding of spiking neural networks and image processing. I discovered famous methods that are a crucial part of these areas and even implemented a major part of them. Secondly, I had the ability to dive into the neuroscience field of study and understand the human nervous system in greater detail. This enabled me to visualize the nerve activity in the brain which facilitates the creation of the network as well as understanding how the visual attention technique works.

In addition, I have gained a set of new skills while working on the project. Researching specific problems enabled me to gather information much faster and boost my searching abilities. The numerous failing experiments and unsuccessful model attempts boosted my patience and taught me how to work under pressure. Writing the dissertation report taught me how to present and analyze data in a more scientific and professional format. Moreover, time management played a key role in the project development especially when unexpected behavior was detected or deadlines were approaching. Finally, I learnt how to optimize the produced code in a way to detect errors much facilely.

Chapter 8
Bibliography

[1] H. Jain, A. Vikram, Mohana, A. Kashyap and A. Jain, "Weapon Detection using Artificial Intelligence and Deep Learning for Security Applications," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 193-198, doi: 10.1109/ICESC48915.2020.9155832.

[2] Self-Driving Cars Technology & Solutions from NVIDIA Automotive. (2021). Retrieved 17 April 2021, from https://www.nvidia.com/en-us/self-driving-cars/

[3] Future of Driving (2021). Retrieved 17 April 2021, from https://www.tesla.com/en_GB/autopilot

[4] Alarcon, N. (2021). DRIVE Labs: Classifying Traffic Signs and Traffic Lights with SignNet and LightNet DNNs | NVIDIA Developer Blog. Retrieved 17 April 2021, from https://developer.nvidia.com/blog/drive-labs-signnet-and-lighnet-dnns/

[5] Daniel Hernández García, Samantha Adams, Alex Rast, Thomas Wennekers, Steve Furber, Angelo Cangelosi, Visual attention and object naming in humanoid robots using a bio-inspired spiking neural network, Robotics and Autonomous Systems, Volume 104, 2018, Pages 56-71, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2018.02.010. (https://www.sciencedirect.com/science/article/pii/S0921889017302439) Keywords: Neurorobotics; Object naming; Visual attention; Biological inspired models; Spiking neural networks

[6] J.A. Hofheimer, B.M. Lester, Neuropsychological Assessment, Editor(s): Marshall M. Haith, Janette B. Benson, Encyclopedia of Infant and Early Childhood Development, Academic Press, 2008, Pages 425-438, ISBN 9780123708779, https://doi.org/10.1016/B978-012370877-9.00110-9. (https://www.sciencedirect.com/science/article/pii/B9780123708779001109) Keywords: Assessment; Development; Fetus; Infant; Neurobehavior; Neuropsychology; Newborn; Risk; Toddler

[7] Building, N. (2021). About Neuroscience - Department of Neuroscience. Retrieved 23 April 2021, from https://neuro.georgetown.edu/about-neuroscience/

[8] The not extraordinary human brain, Suzana Herculano-Houzel, Proceedings of the National Academy of Sciences Jun 2012, 109 (Supplement 1) 10661-10668; DOI: 10.1073/pnas.1201895109

[9] Dayan, Peter. Theoretical neuroscience : computational and mathematical modeling of neural systems / Peter Dayan and L.F. Abbott. p. cm. – (Computational neuroscience) Includes bibliographical references. ISBN 0-262-04199-5 (hc. : alk. paper) — 0-262-

54185-8 (pb.) 1. Neural networks (Neurobiology) – Computer simulation. 2. Human information processing – Compute

[10] Hanson, J. (2021). *Nerve Repair and Acupuncture* [Image]. Retrieved from https://doctorhanson.com/wp-content/uploads/2019/12/nerve-reapir-and-acupuncture.png

[11] Tobimatsu, S. (2012). *The parallel visual pathways in humans.* [Image]. Retrieved from https://www.researchgate.net/profile/Shozo-Tobimatsu/publication/51505859/figure/fig1/AS:202879815163913@1425381725803/The-parallel-visual-pathways-in-humans-Abbreviations-in-this-and-subsequent-figures-d-d.png

[12] Hubel D., Eye, Brain, and Vision, Scientific American Library Series, Henry Holt and Company (1995)

[13] Faruqui, N. (2019). *What is Kernel in Image Processing?* [Image]. Retrieved from https://www.nzfaruqui.com/wp-content/uploads/2019/03/edge-detection-using-kernel.png

[14] Song S., Miller K.D., Abbott L.F., Competitive Hebbian learning through spike-timing-dependent synaptic plasticity, Nature Neurosci., 3 (2000), pp. 919-926

[15] *Maximum pooling*. [Image]. Retrieved from https://d2l.ai/_images/pooling.svg

[16] Network structure of neurons with synaptic delays. (2019). [Image]. Retrieved from https://www.frontiersin.org/files/Articles/429572/fnins-13-00252-HTML/image_m/fnins-13-00252-g002.jpg

[17] Richard Leach, Chapter 8 - Surface Topography Characterisation, Editor(s): Richard Leach, In Micro and Nano Technologies, Fundamental Principles of Engineering Nanometrology (Second Edition), William Andrew Publishing, 2014, Pages 241-294, ISBN 9781455777532, https://doi.org/10.1016/B978-1-4557-7753-2.00008-6. (https://www.sciencedirect.com/science/article/pii/B9781455777532000086) Keywords: Surface Topography Characterisation; Profile Characterisation; Areal Characterisation; Filtering; Surface Texture Parameters; Field Parameters; Feature Parameters; Fractal Methods

[18] S.V. Adams, A.D. Rast, C. Patterson, F. Galluppi, K. Brohan, J.-A. Pérez-Carrasco, T. Wennekers, S. Furber, A. Cangelosi, Towards real-world neurorobotics: Integrated neuromorphic visual attention, in: Neural Information Processing: 21st International Conference, ICONIP 2014, 2014, pp. 563–570.

[19] TensorFlow. (2021). Retrieved 23 April 2021, from https://www.tensorflow.org/

[20] Team, K. (2021). Keras: the Python deep learning API. Retrieved 23 April 2021, from https://keras.io/

[21] PyTorch. (2021). Retrieved 23 April 2021, from https://pytorch.org/

[22] PyNN reference, Davison AP, Brüderle D, Eppler JM, Kremkow J, Muller E, Pecevski DA, Perrinet L and Yger P (2009) PyNN: a common interface for neuronal network simulators. *Front. Neuroinform.* 2:11 doi:10.3389/neuro.11.011.2008

[23] NEST - NeuralEnsemble. (2021). Retrieved 23 April 2021, from https://neuralensemble.org/NEST/

[24] miladmozafari/SpykeTorch. (2021). Retrieved 23 April 2021, from https://github.com/miladmozafari/SpykeTorch

[25] Unsplash Dataset | The world’s largest open library dataset. (2021). Retrieved 23 April 2021, from https://unsplash.com/data

[26] CBCL StreetScenes Database Download Page:. (2021). Retrieved 23 April 2021, from http://cbcl.mit.edu/software-datasets/streetscenes/

[27] (2021). Retrieved 23 April 2021, from https://miro.medium.com/max/869/1*nGHLq1hx0gt02OK4l8WmRg.png